# 下津ゼミ　DPlecture 第2回

金子　雄祐

2014/11/26

# 目次

1. 理論編
2. 数値計算編
3. 計量編

# 2.0　はじめに

- DP の数値計算についての話題を扱う
- Rust（1987）が読める程度の知識を身につけることが目標
- まず標準的な DP についての数値計算の話題を扱う。
- stochastic DP のコードは省略

# 2.1 Methods

- 前回に 2 つの手法を紹介した;Value Function iteration & Policy Function iteration
- 一般的には Policy Function iteration の方が計算が早いとされる
- まず, 方法の説明の前に PC での数値計算の際に重要な点について解説する

# 2.1 Methods

- 実際に数値計算によって求める際にはPCの負荷を軽減することが重要になる
- PCへの負担が大きすぎると計算に時間がかかりすぎる
- また、状態変数や操作変数が連続だと計算不可能になる
- 近似法として離散近似法がある
- これは、状態変数などをいくつもの点の集合として近似する方法で連続微分可能性を仮定しなくて良いので一般的(後で実際にコードを見るときに詳述)

# 2.1 Methods

- Matlab か Python を使用すると先週予告したが予定を変更 (どっちもは無理でしたすいません)
- Matlab と高い互換性のある Octave を使う
- Matlab は高価で利用しづらいという一面があるが、Octave は無料。しかし Octave は Matlab より遥かに遅い（らしい）
- インストール法は下記サイトを参照。また、エディタはなんでも良いが Notepad++を私は使用した
  http://www.frad.t.u -tokyo.ac.jp/public/octave/octave_install.html.ja

# 2.1 Methods

- 余談だが Python を使用したいなら Anaconda というパッケージがある (数値計算用の numpy などがセット)
- Quantative economics など、Python を用いた数値計算についてのサイトは近年充実してきている感があるので Python の方が良いかもしれない（無料だし Octave より速いらしい）
- 導入するならば日本語のサイトなら尾山ゼミのサイトなどを見ると良い

# 2.1 Methods

Value Function iteration

- 前回スライドのケーキ消費問題を例に取る。
- 価値関数が一定の値まで収束するまでベルマン方程式を繰り返し解き続けることになる

# 2.1 Methods

Value Function iteration

- なんでも良いのでとりあえず $V_0$ の形を推測する (例)

$$V_0(x_t) = 1$$

- 次にベルマン方程式に代入する

$$V_1(x_t) = \max_{x_{t+1} \in [0, x_t]} u(x_t - x_{t+1}) + \delta V_0(x_{t+1})$$

Value Function iteration

- $V_1$ の値を再びベルマン方程式に代入する

$$V_2(x_t) = \max_{x_{t+1} \in [0, x_t]} u(x_t - x_{t+1}) + \delta V_1(x_{t+1})$$

- これを V が収束するまで繰り返す。
  $(|V_j - V_{j-1}| \leq \epsilon)$
- 実際に値は収束するのか？

# 2.1 Methods

Value Function iteration

- 先ほどのプロセスを書き直すと以下のようになる

$$\boldsymbol{V_1 = u + \delta V_0}$$
$$\boldsymbol{V_2 = u + \delta[u + \delta V_0]}$$
$$\boldsymbol{V_J = \sum_{j=0}^{J} \delta^T u + \delta^J V_0}$$

# 2.1 Methods

Value Function iteration

$$\boldsymbol{V_1 = u + \delta V_0}$$
$$\boldsymbol{V_2 = u + \delta[u + \delta V_0]}$$
$$\boldsymbol{V_J = \sum_{j=0}^{J} \delta^T u + \delta^J V_0}$$

- $\boldsymbol{\delta^J V_0}$ は 0 に収束していき、$\boldsymbol{V_0}$ での誤った推測の誤差は消滅、求めたい $\boldsymbol{V^*}$ が求まる (本当は Contraction Mapping Theorem の話などをしなくてはいけないが省略。SL 参照)

# 2.1 Methods

Value Function iteration

- なるべく $V^*$ に近い $V_0$ を最初に推測したほうが当然計算時間は短くなる
- 最適成長モデルを例に実際のコードを説明する

# 1.2 有限期間最適化問題

Optimal growth model

- 家計効用最大化問題
  $\sum_{t=1}^{\infty} \beta^{t-1} u(c_t)$
- 家計制約は以下の式
  $y_t = c_t + i_t$
  $y_t; income\ c_t; consumption\ i_t; investment$
- 資本ストックの式は次
  $k_{t+1} = k_t(1-\delta) + i_t$
- Value function は次式 (for all k)

$$\max_{k'} \; V(k) = u(f(k) + (1-\delta)k - k') + \beta V(k'$$

# 1.2 有限期間最適化問題

Optimal growth model

- 先ほども述べたように、実行可能な $k'$ についての離散な点の集まり（grid）を作らなくてはいけない
- $k'$ の範囲の求め方については定常状態まわりに取れば良い (定常状態に至る経路が社会的に最適になるから。Core MacroII）
- また、最終的に得られる解から、Value Function は次のような形になる
  $V_{i+1}(k) = max\ u + \beta V_i(k')$
  $V^* = u/(1-\beta)$

# 1.2 有限期間最適化問題

Optimal growth model

- 具体的計算のため関数を以下のように仮定する
  $u(c) = \frac{c^{1-\delta}-1}{1-\delta}$
  $f(k) = k^{\alpha}$
  $k' = k^{\alpha} - c(1-\delta)k$
- この結果、ベルマン方程式は次のようになる

$$\max_{0 \leq c \leq k^{\alpha}+(1-\delta)k} V(k) = \frac{c^{1-\delta}-1}{1-\delta} + \beta V(k')$$

# 1.2 有限期間最適化問題

Optimal growth model

- c について書き直すと以下のようになる
  $c = k^{\alpha} + (1-\delta)k - k'$
- この結果、ベルマン方程式を書き直すと、

$$V(k) = \frac{(k^{\alpha} + (1-\delta)k - k')^{1-\delta} - 1}{1-\delta} + \beta V(k')$$

$$\left( \max_{(1-\delta)k \leq k' \leq k^{\alpha} + (1-\delta)k} \right)$$

# 1.2 有限期間最適化問題

Optimal growth model

- プログラムを回していく
- パラメータは予め定める
- grid を細かくし過ぎると計算に時間がかかりすぎる
- 収束の許容範囲 ($\epsilon$) は定めておく (Lec1.m のコード見ながら解説)

# 1.2 有限期間最適化問題

Interpolation

- Lec1.m の方法だと時間がかかりすぎる
- Value function iteration の時間削減の方法として Interpolation という方法がある
- これは、状態変数でなく操作変数にも grid を入れる方法
- それ以外はやることはほぼ変わらない
- Lec2.m のコードを見ながら解説

# 1.2 有限期間最適化問題

Policy Function iteration

- Value Function Iteration の時同様、状態変数に grid を導入、効用関数形は特定化する
- 次に候補として政策関数 ($y = f_0(x)$) を推測し、それを価値関数に代入する
  $V_(x_t) = \sum_{s=0}^{\infty} \beta^s u(f_i(x_{t+s}), x_{t+s})$
- 次にこれによって得られた価値関数を元に新たな政策関数を求める

$$f_1 \in \underset{y}{\text{Argmax}} u(y, x) + \beta V(x')$$

# 1.2 有限期間最適化問題

Policy Function iteration

- これによって求められた新たな政策関数を再びベルマン方程式に代入し、というプロセスを政策関数が収束するまで繰り返す
- 具体例を見たほうが分かりやすいので Lec3.m を見ながら説明

# 1.4 DP(non stochastic)

- Esitimation についての話題に入る
- 前回 Estiomation のコードについての話も行うと言ったが話が無駄に難解になるので省略
- Estimation の方法は多岐にわたるので、ここでは Rust が推定の際に用いたアルゴリズムについて非常に大雑把に解説する
- 12 月に元の論文を読むので、ここでは概略を話すに留め、詳しい話はしない
(興味があるならば AC の chap.4 を参照)

# 1.4 DP(non stochastic)

- 最適化を行う本人には分かるが分析者には分からない観察不可能なパラメーター(浅野中村でも10章でも扱った潜在変数)が導入される
- ケーキの例では嗜好の変化,Rustではエンジニアの主観的なコスト（バスが壊れた時に失われる乗客と信用のコスト）である

# 1.4 DP(non stochastic)

Rust framework(大雑把に)

- ステート $x_t$ をそのエンジンでの累積マイル、$i_t = 0, 1$ をエンジン交換の質的変数（1 なら交換）とする。
- 今期から次期へのマイル数の変化に iid な指数分布を仮定して遷移確率 $p(x_{t+1}|x_t, i_t, \theta_2)$ と表現（それなら平均と標準偏差はほぼ等しくなるのでデータからは明らかに iid な指数分布じゃないが）
- このモデルなら政策関数は累積マイルがある閾値を超えたらエンジン交換という形で与えられる
- しかし、交換時の累積マイルのばらつきからデータからはある閾値を超えて変更しているとは考えにくい

# 1.4 DP(non stochastic)

Rust framework(大雑把に)

- 効用関数に付加的な誤差項 ($\epsilon$) を unobservable な状態変数としてモデルを解く
- 遷移確率にこの誤差項も加える
- $\epsilon$ が連続だと計算の時に困るので grid を導入する
- 条件付きの選択の確率を求めなくてはいけない
- 次の仮定を導入する
  $$p(x_t t+1, \epsilon_{t+1} | x_t, \epsilon_t, i, \theta_2, \theta_3)$$
  $$= q(\epsilon_{t+1} | x_{t+1}, \theta_2) p(x_{t+1} | x_t, i.\theta_3)$$

# 1.5 DP(stochastic)

- $P(a|x;\theta)$ は次の多項ロジットモデルで与えられる $P(a|x;\theta) = \frac{exp[u(a,x;\theta)+EV_\theta(a,x)]}{\sum_{j=1}^{J} exp[u(j,x;\theta)+EX_\theta(j,x)]}$
- $\theta$ は最尤法による推定が可能 (浅野中村 10 章)
- Rust はロジットモデルと最尤法から Nested Fixed Point algorithm という推定法を実行した

# 1.5 DP(stochastic)

NFPA

- A. まず、$\theta$ の値の候補（初期状態は guess?) から、Iteration でベルマン方程式を解き、$EV_{\theta}(a,x)$ を計算し、$P(a|x;\theta)$ を計算する
- B. 次に求めた P の値を使い、最尤法から、$N^{-1}\sum_{i=1}^{N} lnP(a_i|x_i;\theta)$ を最大化する $\theta$ を求める
- 得られた $\theta$ から、再び A を実行し、B,A のループを $\theta$ と P が収束するまで繰り返す
- これにより, 得たい $\hat{\theta}$ が得られる

# 1.7 参考文献


- Adda, Jerome, and Russell W. Cooper. Dynamic economics: quantitative methods and applications. MIT press, 2003.
- Stokey, Nancy, and R. Lucas. "with E. Prescott (1989): Recursive Methods in Economic Dynamics." 3.
- Ljungqvist, Lars, and Thomas J. Sargent. Recursive macroeconomic theory. MIT press, 2004.
- Fabrice Collard Lecture Note http://fabcol.free.fr/pdf/lectnotes7.pdf (+ more)